Using Query-Log Based Collective Intelligence to Generate Query Suggestions for Tagged Content Search

Dirk Guijt[1] and Claudia Hauff[2](✉)

[1] Sanoma, Hoofddorp, The Netherlands
dirk.guijt@sanoma.com
[2] Web Information Systems, TU Delft, Delft, The Netherlands
c.hauff@tudelft.nl

Abstract. One of the standard features of today's major Web search engines are query suggestions, which aid the user in the formulation of their search queries. Over the years, a number of different approaches have been proposed which have commonly been evaluated in the standard Web search setting. In this work, we build a query suggestion pipeline based on the collective intelligence stored in log data collected from a more constrained search engine which uses tags to index the content. This constrained environment, though large-scale, differs considerably from standard Web search with respect to its users, indexing process and Web coverage. We implement a number of suggestion approaches based on query-flow and term-query graph models and investigate to what extent they are applicable in this more constrained environment.

Keywords: Query suggestions · Query-flow graphs · Search sessions · Collective intelligence · Tags · Tagged content

1 Introduction

One of the standard features of today's major Web search engines are query suggestions, which aid the user in the formulation of their search queries whilst typing. Those suggestions are commonly generated using the collective intelligence of the search engine users which is stored in the search engine's query logs. Learning the behaviour of the search engine users and applying the knowledge by generating query suggestions is not an easy task, as large-scale query logs are noisy, may contain errors and logging artefacts.

In this work, we present our efforts on implementing query suggestions for `startpagina`[1], a Dutch Web search portal similar in spirit to the Open Directory Project[2]. It is currently relying on query suggestions offered by a major

The work was conducted during the first author's TU Delft Master thesis project at Sanoma.

[1] http://startpagina.nl

[2] Open Directory Project (now DMOZ): http://www.dmoz.org

Commercial Search Engine (CSE) instead of generating its own. Since interacting with those externally generated suggestions leads to general Web search results (which often include results not indexed by `startpagina`) many users leave `startpagina` in the middle of a search session and continue with a general Web search. While search engine switching (initiated by the user) is a well-known phenomenon [25], we consider this switch to be a rather unconscious one, as the users are "switched" to a different engine without their explicit request. In order to increase user retention, we use `startpagina`'s query log and implement a four-step query suggestion pipeline based on state-of-the-art approaches in (1) search session splitting [16,17], (2) the classification of query reformulations [9,19], (3) the generation of query-flow graphs, and, (4) the extraction of suggestions from such graphs [7–10].

Our main research questions focus on the effects of applying provenly effective methods of generating query suggestions in regular Web search engines to a more constrained search environment, which is similar to search on a social tagging portal. In contrast to regular Web search, our experimental environment indexes its contents through tags created by human annotators. Our search engine is limited in its ability to return results for all queries, as a given query must match a tag before the associated content can be returned. It is not useful to suggest queries for which no matching tag exists and thus no content is available. In this work, we investigate whether query logs can be used to generate relevant query suggestions in this context.

We derive six different models from state-of-the-art query-flow and term-query graph models [7–10] and compare their effectiveness to the suggestions provided by the CSE (our baseline that is currently in use at `startpagina`). In this particular use case we are able to achieve much higher coverage, that is, the models trained on our log data can provide valid suggestions, i.e. suggestions for which `startpagina` has results, for a much larger number of queries.

In the remainder of the paper, we first cover related work (§ 2), provide more details about `startpagina` (§ 3) and describe our approach (§ 4), before presenting our experiments (§ 5) and a discussing of the lessons learnt (§ 6).

2 Background

2.1 Search Sessions

Deriving search sessions, i.e. chronologically ordered sets of queries by a single user with a common search goal (within a certain time interval), from query logs is one of the necessary pre-processing steps to generate query-log based query suggestions. Search session splitting has been tackled by various researchers in the past, including [15–17]. Most works rely heavily on lexicographical and temporal properties; it was shown that those low-level features achieve nearly the same effectiveness for search session splitting as more complex semantic or result-set based features [16].

2.2 Query Reformulations

Another important aspect of our pipeline is the classification of query reformulations, i.e. in what manner the next query is altered with respect to the previous one, as the reformulation type itself can significantly alter the shape and dynamics of a query-flow graph [8,9]. Early work on user query behavior in web search defined various reformulation type classes [12,20]. These form the basis for the four reformulation types introduced in [9], which are: *specialisation* (.e.g query “elephant” is reformulated to “elephant tusk”), *generalisation* (e.g. moving from “elephant tusk” to “elephant”), *parallel move/equivalent rephrase* (replacing some terms with similar terms or phrase the same query intent differently), *same query/error correction* (applying small changes, often spelling correction, to the next query). Both Boldi et al. [9] and Huang et al. [19] provide an extensive list and description of features that can be used to detect the reformulation type of two subsequent queries. These include both lexicographical and temporal features.

2.3 Query Suggestion

Suggestions for a given query can be generated from a multitude of sources: document content, taxonomies (e.g. WordNet), query logs or a combination of those. The use of query logs is particularly popular, but at the same time limited to use cases where large-scale query logs are available.

Huang et al. [18] extracted suggestions based on co-occurring query terms in users’ search sessions. They found this approach to outperform a pure document content analysis-based approach (performed over the top-k retrieved documents). Baeza-Yates et al. [2] combined clicks and retrieved documents by only considering those top-k documents that were actually clicked by users; new query terms are suggested based on the analysis of the clicked documents.

The notion of click graphs started to appear in 2000 [6]. A click graph is a bipartite graph consisting of two groups of nodes, query nodes and document nodes (URLs). There is an undirected edge in between two nodes if the document appears in the result set of a query and was clicked as well. Generating such a graph from a query log enables the use of a range of graph theoretic algorithms. In particular random walks on click graphs are a popular strategy to generate query suggestions, e.g. [13,14,21]. The main drawback of click graphs is the large amount of data required to generate high-quality graphs; not every user query yields a click, and not all search portals are frequented by a sufficient number of users to make click graphs a viable option.

This drawback has led to the use of query-flow graphs for query suggestion generation. A query-flow graph contains nodes constructed from the unique set of queries in the query log. These nodes are connected by a directed edge if the queries succeeded one another in a single search session. The weights of the edges correspond to the likelihood of the two queries belonging to the same query chain. This modelling strategy was first proposed in [7]. Numerous extensions have been proposed, for instance, in [8,9], different graphs are constructed by considering only a subset of edges based on reformulation types; in [3,4] the random walk is biased towards the query intent. Altering the graph itself, by adding shortcuts for very likely paths in the graph has been proposed by Anagnostopoulos et al. [1]. More recently, works have also began to extend such graphs with nodes on the term level: instead of considering only a query as atomic unit (i.e. a node in a graph), each unique term within a query is considered to be a node as well. An advantage of such a term-query graph [10] is that the model can generate suggestions for queries that have not been "seen" before, which is not possible in query-flow graphs. The transition of individual query terms has been added to the query-flow graph in Szpektor et al. [24] and Song et al. [23]. Both of these models have a semantic component as well through named-entity recognition and click-based topic extraction respectively. Since users' querying behaviour and the search system's back-end are likely to change over time, [5] investigated the effects of time on the construction of query-flow graphs. They concluded that query-flow graph indeed age due to the change in user interest.

Query-flow graphs and term-query graphs have shown to be a popular and successful approach to query suggestion generation. We investigate two query-flow graph models (the original model [7] and variations of the reformulation type model [8,9]) and one term-query graph model [10] and examine their applicability to our particular use case `startpagina`. We do not consider click graphs, as the amount of click data required is not suitable for our use case.

3 Startpagina

`startpagina` focuses specifically on the Dutch market. The search portal employs human annotators that manually determine whether or not a Web site d (written in Dutch) should be included in its index. For indexing purposes, annotators provide a number of *tags* $\mathcal{T}_{d_i} = \{t_1^i, ..., t_m^i\}$ for each d_i. Users can only retrieve d_i if their complete query matches a tag in $\mathcal{T}_{d_i}$. The entire tag space over all n documents presented in the index is $\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_{d_k}$. `startpagina` currently employs both auto-completion (offered by the employed commercial search engine) and instant-result mechanisms. Each typing action by a user results in one of the following:

- If the user's current query is a tag found in $\mathcal{T}$, the correspondingly tagged results are shown (instant results provided by `startpagina`).
- If the user's current query is not in $\mathcal{T}$, query suggestions are retrieved from the CSE's suggestion API. The suggestions are agnostic to the tag space $\mathcal{T}$, i.e. they are not restricted to terms and phrases that appear in $\mathcal{T}$. A click on any of the suggestions that are not in $\mathcal{T}$ leads the user to a standard Web search result page (SERP) provided by the CSE.

Figure 1 presents an overview of `startpagina`'s user search flow. The outcome on the left (leading the user to a target Web site) is the desired case: the user remains on `startpagina` until the final click. The outcome on the right (showing a SERP) is the case we are tackling in our work: the external suggestions should be replaced whenever possible (i.e. when we can generate suggestions $t_i \in \mathcal{T}$) with startpagina's own suggestions. In cases where this is not possible, startpagina continues to rely on the CSE's suggestions. Overall, generating startpagina internal suggestions will keep more users engaged with the site, which in turn will improve user retention and user satisfaction.

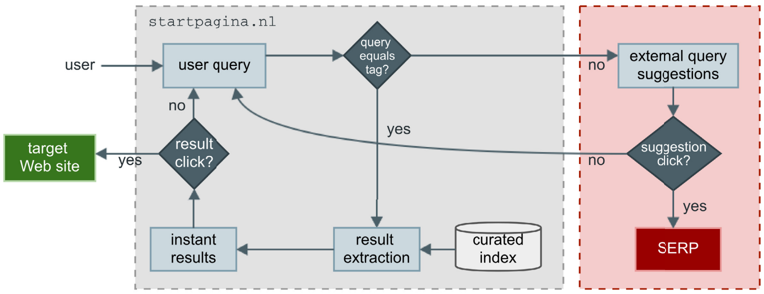

Fig. 1. High-level overview of startpagina's current user search flow. Shown in red (suggestion component) is the part of the search flow we are tackling in this work.

On average, each month startpagina has more than 1.5 million unique visitors who conduct tens of millions of searches. A significant fraction[3] of searches result in the user leaving the site as their queries are not in $\mathcal{T}$ and thus no startpagina results are available.

The goal of our work is to provide for each user query which is not found in $\mathcal{T}$, a list of query suggestions that are found in $\mathcal{T}$. In this manner, when a user submits a search query for which no results are available, she instead receives a number of related query suggestions, all of which will lead to results within startpagina.

The content of startpagina is manually moderated and so are the tags that are used to describe and index it. This manual, human factor is similar to social tagging systems where the content and tags are generated by humans as well. The key difference is that startpagina has an editorial staff as opposed to having the system's users create and curate the content. Although this difference is important to note, our research does not focus on the content creation and curation but on using the collective intelligence of the search engine users to generate query suggestions. We therefore believe that our results extend to search engines that search through folksonomies as well.

4 Approach

Our query suggestions pipeline consists of four components: (1) search session splitting, (2) classification of query reformulation types, (3) generation of

[3] Due to the commercial nature of the portal, the exact numbers cannot be published here.

query-flow graphs, and, (4) deriving suggestions from them. The first two are necessary pre-processing steps to convert the query log into a suitable representation.

4.1 Pre-processing

Search Session Splitting. Given a typical query log (i.e. user-id, timestamp, query and logged click if any), the first pre-processing step involves aggregating the queries submitted by a single user into search sessions. An often used heuristic is to split a search session after 30 or 60 minutes of inactivity [15]. Apart from splitting simply by time, classifiers have been trained in the past [15] to determine whether or not a session boundary exists between two subsequent queries q_i and q_{i+1} submitted by the same user. We opted for this approach as it generally yields better results.

Reformulation Type Classification. One of the query-flow graph models we investigate is based on the intuition that different graphs should be build for different query reformulation types [9] as not all reformulations are equally useful for the generation of query suggestions. Thus, as a second component we implemented a classifier to determine the reformulation type of a pair of queries issued subsequently within a single search session. We distinguish between the following four types [9]: specialisation (**S**), generalisation (**G**), same query/error correction (**C**), and, parallel move/equivalent rephrase (**P**).

We treat both tasks (search session splitting and reformulation type classification) analogously, as they are similar in nature. We derive twenty-three features based on q_i and q_{i+1} individually as well as their temporal and syntactic relationship, in line with previous works [9,16,17,19]. An overview of these features is shown in Table 1. We employ a decision tree classifier (C5.0) as suggested in [9]. For search session splitting our classifier is binary, while for reformulation type classification we use n binary classifiers as well as the n-class classifier (where n is the number of reformulation types). The n binary classifiers are ordered according to their accuracy. If a query pair is classified with sufficient confidence by a classifier in the cascade, the remaining classifiers in the cascade are skipped for that pair [16].

To train the session splitting classifier, we manually annotated a set of randomly drawn *super-sessions* (a session of a single user split by the 60 minute inactivity heuristic is one super-session) from our query log and determined for each pair of subsequent queries whether or not they belong to the same session. Having trained the classifier, we then identified the sessions across the entire query log. In a second step we drew a sample of the search sessions identified in this manner and manually annotated the type of reformulation occurring. To compare our results directly with existing research, we also applied our implementation to a search session data set provided in [16] (we refer to it as `Hagen13`), which is based on the AOL query log.

We evaluate the effectiveness of the classifier through 5-fold cross-validation and report the following four metrics:

Table 1. Description of the 23 features used in both session splitting an query reformulation type classification. The following notation has been used in this table: $q^{pfx} \sqsubset q$ means that q^{pfx} is a prefix of q, q^{url} means that all URL elements like *http://* and *.com* have been stripped from q, and, $w(q)$ is the set of words in query q where the space character was used as a boundary between words.

#	Notation	Feature
$F1$	Δ_{ms}	time difference between q_i and q_{i+1} in milliseconds
$F2$	$\|q_i\|$	number of characters in q_i
$F3$	$\|q_{i+1}\|$	number of characters in q_{i+1}
$F4$	$F3 - F2$	difference in length between q_i and q_{i+1}
$F5$	$\texttt{Equals(}q_i\texttt{, }q_{i+1}\texttt{)}$	boolean equality of q_i and q_{i+1}
$F6$	$\texttt{Equals(}q_i^{pfx}\texttt{, }q_{i+1}\texttt{)}$	boolean equality of q_i^{pfx} and q_{i+1}, with $\|q_i^{pfx}\| = \|q_{i+1}\|$
$F7$	$\texttt{Equals(}q_i\texttt{, }q_{i+1}^{pfx}\texttt{)}$	boolean equality of q_i and q_{i+1}^{pfx}, with $\|q_i\| = \|q_{i+1}^{pfx}\|$
$F8$	$\texttt{Equals(}q_i^{url}\texttt{, }q_{i+1}^{url}\texttt{)}$	boolean equality of q_i^{url} and q_{i+1}^{url}
$F9$	$\|w(q_i)\|$	number of words in q_i
$F10$	$\|w(q_{i+1})\|$	number of words in q_{i+1}
$F11$	$F10 - F9$	difference of number of words in q_i and q_{i+1}
$F12$	$\|w(q_i) \cup w(q_{i+1})\|$	number of unique words combined
$F13$	$\|\{x \in w(q_i), y \in w(q_{i+1}) \| x \sqsubset y\}\|$	number of words that were expanded
$F14$	$\|\{x \in w(q_i), y \in w(q_{i+1}) \| y \sqsubset x\}\|$	number of words that were contracted
$F15$	$\|w(q_{i+1}) - w(q_i)\| - F10 - F11$	number of added words
$F16$	$\|w(q_i) - w(q_{i+1})\| - F10 - F11$	number of deleted words
$F17$	$\texttt{Lev(}q_i\texttt{,}q_{i+1}\texttt{)}$	Levenshtein distance between q_i and q_{i+1}
$F18$	$\frac{F17}{0.5 \cdot (F2+F3)}$	Levenshtein distance normalized by average length
$F19$	$\frac{F17}{F1}$	Levenshtein distance normalized by time difference
$F20$	$\texttt{CosSim(}\theta_{word,i}\texttt{, }\theta_{word,i+1}\texttt{)}$	cosine similarity of character 3-grams extracted from the individual words in a query
$F21$	$\frac{F20}{F1}$	cosine similarity ($F20$) normalized by time
$F22$	$\texttt{CosSim(}\theta_{query,i}\texttt{, }\theta_{query,i+1}\texttt{)}$	cosine similarity of the character 3-grams extracted from the query string as a whole
$F23$	$\frac{F22}{F1}$	cosine similarity ($F22$) normalized by time

- **Correctness**: the percentage of sessions that are completely correct, i.e. the start and end query are correctly identified and no split is introduced in-between;
- **Precision**: the percentage of boundaries that were correctly identified from all the boundaries that where classified as boundaries;
- **Recall**: the percentage of boundaries that were correctly identified of all the boundaries that should have been identified; and,
- **F-Measure**: with $\beta = 1.5$, as we consider false positives to be less harmful then false negatives (in the later case different search sessions are merged into one, leading to a degradation of the query-flow graph).

4.2 From Query-Flow Graphs to Query Suggestions

Query-Flow Graph. Boldi et al.'s query-flow graph [7–9] is a weighted, directed and annotated graph $G_{orig} = (V, E, W, T)$. The set of nodes V is defined as $V = V_{query} \cup \{t\}$ with V_{query} being the set of unique queries in a query log and t being a special node denoting the end of a search session. The set of edges $E \subseteq V \times V$ is a subset of all possible edges. An edge appears between nodes u and v if u was reformulated into v in at least one session in the query log. The edge (u, t) appears when u is the last query in a session. Edges are associated with weights, i.e. $w(u, v) = (0..1] \in W$. The weight is dependent on the frequency of the transition in the query log. Let $OUT(u)$ be the set of nodes that have an incoming edge from u. For every node it holds that: $\sum_{v \in OUT(u)} w(u, v) = 1$, which makes the graph essentially a Markov chain. Then, $w(u, v) = \frac{r(u,v)}{\sum_{i \in OUT(u)} r(u,i)}$; where $r(u, v)$ is the number of times that the transition from u to v occurred in the log. Finally, T is the set of annotations for each edge in E: each edge is annotated with the type of reformulation occurring between u and v. Edges directed to end node t are marked with special type X. Note that the original model did not have any annotations.

Slicing the Query-Flow Graph. As already mentioned, prior work [8,9] has investigated the use of a limited set of reformulation types to generate a more refined version of the query-flow graph. To create a slice of G_{orig}, we only retain those edges that are annotated with our chosen reformulation types; as an example, G_{SP} is the slice of G_{orig} which only contains reformulations of types specialisation and parallel moves. The edge weights are normalised so that $\sum_{v \in OUT(u)} w(u, v) = 1$. The special reformulation type X is assumed to be present in every slice.

Term-Query Graph. Another model that we consider is the term-query graph-based model [10], which again is an extension of G_{orig}. Typically, queries consist of a number of terms. In this model, we enlarge the graph to also add the set of unique terms of the query log as nodes, that is, $V = V_{query} \cup V_{term} \cup \{t\}$. The edge set E is also enlarged: there is an edge between a node $u \in V_{term}$ and $v \in V_{query}$ if v contains the term u. Note that no edges are directed towards term nodes and no direct edges exist between two term nodes. For a node $u \in V_{term}$ and $v \in V_{query}$, it holds that: $w(u, v) = \frac{1}{|OUT(u)|}$. Lastly we note that the reformulation type of an edge (u, v) with $u \in V_{term}$ and $v \in V_{query}$ is the special type X.

Generating Suggestions. Having discussed the different query-flow and term-query graphs, we now turn to the generation of query suggestions. We formulate this problem as follows: given a query q and a graph model G, generate a set of (usually no more than 5) suggestions that are relevant to q's intent.

It is important to note that if G is a query-flow graph model (either G_{orig} or $G_{reformulationTypes}$), we can only generate suggestions for "seen" queries, i.e. queries that are present in the query log from which G has been generated. If a query does not appear in the query log, we cannot generate suggestions for it. If q appears in G, we perform a random walk with restart from q's node, known as Personalized PageRank (PPR) [22]. The difference between PRP and standard PageRank [11] is that in PRP the random jump always jumps to our starting node, as opposed to jumping to a random node in the graph. The result of the random walk is a probability distribution (PPR scores), i.e. the probability of walking across a particular node in G when starting the walk in q. This is in effect the probability of users' submitting a particular query after the initial query q has been submitted. Since in this setup queries that occur very frequent in the query log have a large number of incoming edges with high weights, we normalise the PPR score by the standard PageRank score that each query (node) has. We then filter out all the suggestions that are not in tag space $\mathcal{T}$, with the exception of special node t. The normalised PPR score is then used to rank the queries (nodes) and the highest scoring queries are selected as suggestions for our starting query q. The rank of node t in our generated ranking is also of importance: all queries (nodes) ranked below t have a lower likelihood of being submitted after q than the user stopping her search session. Thus, these queries are considered not to be useful and are stripped from the suggestion list (as well as t itself).

When G is the term-query graph, we first have to split the initial query q into $w_1, ..., w_m$ individual terms. We perform m random walks with restart, each one starting at one of the w_i (if they exist in the graph). The random walks are conducted in the same manner as described for the query-flow graphs. The result is m probability distributions, one for each of q's terms. We then calculate the Hadamard product [10] of these probability distributions, which assigns only to those suggestions that appear in all m distributions a score greater than zero. The suggestions are ranked by their Hadamard score and as before suggestions ranked below node t are removed from the final list of suggestions.

We experiment with eight models, five based on the query-flow graph and different reformulation types, one based on the term-query graph and two relying on the current system's suggestions that are generated by a commercial search engine.

Specifically, we investigate:

- $\mathbf{M_{orig}}$: basic query-flow graph including all reformulations types;
- $\mathbf{M_{GPC}}$: query-flow graph with generalis., parallel move and error correction;
- $\mathbf{M_{GPS}}$: query-flow graph with generalis, parallel move and specialis.;
- $\mathbf{M_{GCS}}$: query-flow graph with generalis, error correction and specialis.;
- $\mathbf{M_{PCS}}$: query-flow graph with parallel move, error correction and specialis.;
- $\mathbf{M_{TGM}}$): the term-query graph model including all reformulations types;

– **CSE suggestions overall**: we retrieve the top-5[4] suggestions for each test query from the system's currently employed CSE, independent of whether or not the suggestions are in $\mathcal{T}$ (i.e. leading to results in `startpagina`);
– **CSE suggestions** $\in \mathcal{T}$: we use the suggestions from the previous model (CSE suggestions overall) as a starting point and filter out all those that do not appear in tag space $\mathcal{T}$ (i.e. those that do not lead to results within `startpagina`).

Training and Evaluation. To train and evaluate our suggestions, we split our query log (ordered by time) into two sets, equivalent to "seen" and "unseen" queries. We generate the query-flow and term-query graphs on the seen set. We sample queries from the unseen set and derive query suggestions for them employing the eight models just described. For evaluation purposes, we ask human evaluators to judge the usefulness of the presented query suggestions to the initial query q as either *Useful*, *Somewhat Useful*, *Not Useful* or *Don't know*. The overall effectiveness of a model is the percentage of queries, for which the top-5 suggestions contain at least one *Somewhat Useful* suggestion (the so-called *u-score*) [8–10]. Additionally, we also evaluate the models according to their *coverage*, i.e. the number of queries for which suggestions are made at all. Since in our setup the tag space $\mathcal{T}$ is limited, the *u-score* alone is not sufficient to evaluate the effectiveness of our models.

5 Experiments

5.1 Data Set

We extract two data sets from `startpagina`'s query log: (1) data set `SP1` contains tens of millions of queries sampled from `startpagina`'s query log collected between April 14, 2014 and June 13, 2014; (2) data set `SP2` contains all queries issued in the seven days starting at June 14, 2014. We use `SP1` to train our classifiers and models. Data set `SP2` is employed to evaluate the query suggestions generated by our pipeline - this is a realistic setup, as we may be only able to (re-train) our models at particular moments in time.

5.2 Pre-processing

We randomly selected 974 super-sessions (i.e. a session split after 60 minutes of inactivity) from `SP1` and manually annotated the session boundaries within those super-sessions. Overall, we were able to unambiguously determine boundaries for 936 super-sessions, yielding 2,489 different search sessions with a total of 9,410 queries.

Our search session splitting classifier determines for each pair (q_i, q_{i+1}) of subsequent queries (submitted by a single user) whether they belong to the

[4] The API of the employed CSE imposes this limit.

same session or not. If not, q_i is considered to be the last query of session S_j while q_{i+1} is the first query of search session S_{j+1}. We use 5-fold cross-validation for training and testing.

The results are presented in Table 2. As baseline we report search session splitting based on minutes of inactivity (for each dataset, we empirically evaluated inactivity cutoffs between $[1, 2, .., 60]$ minutes and report the best performing one). While the inactivity based approach leads to 68% fully correctly identified sessions for SP1, our classifier-based approach makes correct decisions for 83% of the sessions. This result is verified on the public Hagen13 data set. When considering the learnt decision trees, the most significant features are the cosine similarity (both on the word and query level) as well as the Levenshtein distance normalised by the queries' time difference (i.e. $\frac{L(q_i,q_{i+1})}{\Delta_s}$). Overall, we conclude that we are able to split the Dutch startpagina sessions with sufficient accuracy for our purposes.

Table 2. Overview of search session splitting results achieved on two data sets

Splitting method	**Data set**	**Correct**	**Precision**	**Recall**	**F$_{1.5}$-Measure**
Inactivity (2 min)	SP1	68.46%	0.65	0.95	0.84
Classifier (C5.0)	SP1	82.64%	0.79	0.99	0.92
Inactivity (11 min)	Hagen13	67.35%	0.83	0.88	0.86
Classifier (C5.0)	Hagen13	84.51%	0.82	0.99	0.93

Having trained our search session splitter, we ran the classifier on the entire SP1 data set. We then randomly selected 1,806 query pairs from the identified sessions. For 1,738 pairs we were able to manually annotate them with their respective reformulation type (the remaining query pairs had unclear reformulations). We used those pairs to train and evaluate our reformulation type classifiers in a 5-fold cross-validation setup. The results show that query type reformulations can be classified effectively, yielding an accuracy of 85.9%.

Having determined the effectiveness of our classifier cascade, we classified all query pairs in SP1. We list the distribution of identified reformulation types in Table 3.

Interestingly, compared to previous works, in particular [9], which use Yahoo! query log data from 2008, we find considerable differences for the parallel move and specialisation types: parallel moves are found in less than 20% of all cases in SP1, while it is the most prevalent reformulation type in the Yahoo! logs (roughly 50% of all reformulations). In contrast, we find that in SP1 the S and C types occur with considerably higher frequency than found in the standard Web search setting. Despite the fact that for many queries startpagina does not return any results (because the query does not appear in $\mathcal{T}$), more than 50% of the reformulations lead to more specific queries. This is contrary to our intuition, as we would expect users to back-off to more general queries in order to receive startpagina results. More research is needed to investigate this behaviour.

Table 3. Overview of the types of reformulations found in data set SP1 and in previously published works [9] (final two columns)

Reformulation type	SP1 $n = 4$	**Yahoo! UK, 2008**	**Yahoo! US, 2008**
G	13.01%	4.40%	9.50%
P	16.26%	47.70%	55.50%
C	18.16%	10.40%	5.00%
S	52.56%	37.50%	30.10%

5.3 From Query-Flow Graphs to Query Suggestions

We built the query-flow graphs from SP1 using our implemented pipeline: the query log was split into sessions, the reformulations were annotated, duplicate queries in succession were removed, and for each of the models a query-flow graph was generated. When computing (personalized) PageRank we set the random jump probability to $(1-\alpha) = 0.15$. Some basic graph statistics are shown in Table 4. While for most induced graphs the number of nodes and edges is comparable, $\mathbf{M_{GPC}}$ and $\mathbf{M_{TGM}}$ differ considerable with respect to the maximum out-degree and the number of edges. In slice $\mathbf{M_{GPC}}$, all edges that represent a specialization have been removed. We have shown in Table 3 that over half of the reformulation types in the graph are specializations so we expect the maximum outdegree to be lower for this slice. The border case nodes with a high outdegree have an even higher percentage of (outgoing) specialization edges, which explains the lower maximum outdegree in the overal graph. The higher number of nodes and edges in model $\mathbf{M_{TGM}}$ can be explained by the addition of the term nodes and the edges from the term nodes to the query nodes.

Table 4. Overview of the generated query-flow and term-query graphs based on data set SP1

	#nodes	**#edges**	**max. outdegree**
$\mathbf{M_{orig}}$	14,153,454	19,611,085	13,699
$\mathbf{M_{GPC}}$	14,153,454	12,638,766	775
$\mathbf{M_{GPS}}$	14,153,454	17,271,592	13,643
$\mathbf{M_{GCS}}$	14,153,454	16,697,988	13,650
$\mathbf{M_{PCS}}$	14,153,454	18,139,039	13,696
$\mathbf{M_{TGM}}$	17,686,843	59,658,990	552,735

We drew test queries from SP2 as follows: we partitioned SP2 into search sessions and randomly selected 282 of them. From each session, we then randomly drew a single query and generated the top 5 query suggestions based on each of our six models. We also captured the top 5 query suggestions from the system's currently used CSE. On average, each session contained 2.9 ($\sigma = 1.65$) queries.

Table 5 contains an example of query suggestions generated for the first query (underlined) of the following search session consisting of two queries: {*egon* → *aegon*}. Aegon is a multinational life insurance, pensions and asset management company. The user misspelled the name of the company in the first query. The final column lists the rating our human annotators assigned to the suggestion when taking the whole context of the search session into account.

We merged all suggestions generated for a query into a single list, removing duplications; due to the overlap in generated suggestions between the different models, the final list had far fewer than the possible $5 \times 7 = 35$ suggestions. On average 9.85 ($\sigma = 3.49$) unique suggestions were generated per query.

Table 5. Example of query suggestions generated for a query. From the short search session {*egon* → *aegon*} the first query (*egon*) was selected as the query to generate suggestions for. For each generated suggestion, all models generating it in the top 5 results are listed. Query suggestions that do not appear within $\mathcal{T}$ are marked with †. The final column contains the rating our human annotators: either U (*Useful*), US (*Somewhat useful*), or, NU (*Not useful*).

Query	**English**	**Model(s)**							**Rating**
suggestions	**translation**	Orig	GCS	GPC	GPS	PCS	TGM	CSE	
aegon	aegon	x	x	x		x	x		U
egon zehnder †	egon zehnder							x	NU
aegon hypotheek	aegon mortgage	x	x	x		x			U
aego	aego			x					NU
egon schiele †	egon schiele							x	NU
mijnaegon	myaegon			x					U
mijn aegon	my aegon				x		x		U
egon †	egon							x	NU
inloggen aegon	login aegon						x		U
aeg	aeg			x					SU
egon derksen †	egon derksen							x	NU
aegon verzekering	aegon insurance	x			x	x	x		U
aegon.nl	aegon.nl	x	x		x	x			U
aegon inloggen	aegon login	x	x			x	x		U
egon krenz †	egon krenz							x	NU

We recruited 31 native speakers of Dutch as human evaluators (11 human annotators employed at `startpagina` and 20 `startpagina` users). The evaluators were presented with a search session and the selected query (to derive suggestions for) from that session. To ensure that our evaluators made high-quality judgements, we asked them two initial questions for each session/query pair: (1) do all queries in the session have the same search goal (*yes/somewhat/no*), and, (2) do you understand the user's search intent (*yes/somewhat/no*). Each session/query pair was judged by a single evaluator. An evaluator could judge up to 20 session/query pairs. From the 282 session/query pairs, 40 sessions were judged by our evaluators as not having a single goal, while for an additional 53 sessions the evaluators were not able to clearly identify the intent of the session. For the remaining 189 session/query pairs with a total of 1886 suggestions, the evaluators were asked to rate the list of distinct suggestions with respect to their usefulness. The suggestions were presented in random order and the evaluators were unaware of the model(s) that generated the suggestion. In Table 6 we present the different models' effectiveness with respect to u-score and coverage.

Let's first discuss the influence of the reformulation types, when comparing the five reformulation type models (ignoring M_{TGM}) generated based on `SP1`. **S** & **G** are both important for coverage. This is not unexpected, as **S**-class reformulations account for more than 52% of all reformulations (Table 3). More interesting is the influence of the **G**-class edges. Although they account for only 13% of all reformulations (less than **P** or **C**), they are essential to achieve high coverage ($\mathbf{M_{orig}}$ achieves 88% coverage, $\mathbf{M_{PCS}}$ only 56%). We reason that `startpagina`'s annotations play a role here: complex long queries rarely appear in $\mathcal{T}$, and thus simpler queries that generalise (i.e. are shorter and may appear in $\mathcal{T}$) are valuable. With respect to the u-score, i.e. the usefulness of the generated suggestions, we find that *including* the **P**-class (i.e. parallel moves) has the most negative effect (on average, models that include the P-class lose 10% in u-score). We speculate that parallel moves connect several distinct search paths together, thus adding noise to the reformulations.

The term-graph model ($\mathbf{M_{TGM}}$) exhibits the highest u-score across all our models, i.e. it generates the best suggestions for those queries, for which it is able to generate anything. At the same time though, the coverage is much lower than any of the other models (less than 36%), making it unusable in practice. This model was designed to have a higher coverage for queries that are not seen before. It is therefore notable that the model has the lowest coverage in our evaluation. The reason for this could be due to the limited tag space combined with the term-query model specific random walk method for generating suggestions. The individual lists of suggestions per random walk from a term are already sparse and the overlap between them is expected to be even more sparse (or non-existent). For future work it will be interesting to investigate models that combine reformulation type models with term-query graph models. Unfortunately, this is not feasible for the `startpagina` use case as the coverage for this combination would be very low and almost no suggestions could be made.

Overall, we conclude that when balancing the needs for usefulness and coverage, the basic model, i.e. $\mathbf{M_{orig}}$, which does not distinguish between reformulation types, is the best one to use.

When comparing our models' results to the CSE suggestions we first notice the low coverage CSE suggestions achieve - less than 26% of the individual generated suggestions are tags in `startpagina`'s tag space $\mathcal{T}$. This is not surprising, as the CSE suggestions are agnostic to `startpagina`'s content. With respect to u-score (which ignores whether or not the suggestions are in $\mathcal{T}$), CSE suggestions considerably outperform our models, indicating that there is a large potential for future improvements. Thus, we conclude that although the suggestions provided by the CSE are useful ones, they do not help in finding `startpagina` content most of the time.

Table 6. Overview of the query-flow graph models' effectiveness. All but the *CSE suggestions* are generated using `SP1` as training data.

	u-score	coverage
$\mathbf{M_{orig}}$	62.28%	88.36%
$\mathbf{M_{GPC}}$	63.78%	67.20%
$\mathbf{M_{GPS}}$	60.81%	78.31%
$\mathbf{M_{GCS}}$	70.07%	72.49%
$\mathbf{M_{PCS}}$	60.55%	55.67%
$\mathbf{M_{TGM}}$	72.05%	35.98%
CSE suggestions $\in \mathcal{T}$	83.67%	25.93%
CSE suggestions overall	87.36%	96.30%

When we consider the overal effectiveness of our models to generate *useful* suggestions, we find that more than half of all 1886 suggestions judged were deemed at least somewhat useful by our evaluators; more specifically: 28.2% of suggestions were deemed *useful*, 23.44% where *somewhat useful*, 44.11% were *not useful* and 4.24% suggestions were rated as *don't know*.

For 160 of the 189 queries, both $\mathbf{M_{orig}}$ and **CSE suggestions overall** could generate suggestions. In 75% of these cases, there was no overlap in the top-5 suggestions of both models. This indicates that our investigated approaches relying on query-flow and term-query graphs and `startpagina`'s query log are considerably different from the approaches and data employed by the CSE we compare our work against.

6 Conclusions

In this paper, we implemented and evaluated a pipeline that, based on a query log, generates query suggestions for unseen queries. We built on prior work, and investigated established approaches for search session splitting, reformulation type classification and the generation of query suggestions from query-flow and term-query graphs. We were able to successfully apply the results obtained in a regular Web search setting to the Dutch curated content search environment of `startpagina`. While search session splitting and reformulation classification yielded similar to existing work, the use of query-flow and term-query graphs showed a number of differences:

– We can improve the usefulness score (u-score) of the suggestions when only considering particular types of reformulations, as expected. In our case, leaving out class **P** (parallel move) increases the effectiveness. Contrary to prior work, **S** (specialisation) reformulation edges do not play a significant role.

– In contrast to the literature, which usually focuses on the u-score as effectiveness measure, we also need to take the coverage of a model into account.

Here, we observe a considerable degradation in coverage when only a subset of reformulations is considered; thus, there is a trade-off between accuracy and completeness. Depending on the optimisation goal, different models would have to be chosen.

– The term-query graph model produces the highest u-scores but, surprisingly, also has the lowest coverage by far. We reason that this is caused by the limited tag space $\mathcal{T}$ combined with the use of the term-query model specific random walk method.

Overall, we conclude that the state-of-the-art query suggestion models can successfully be applied to search environments where the content is described and indexed using tags. Not all user queries return results in such an environment but through harnessing the collective intelligence stored in the query logs we are still able to actively support individual users in their search missions. We found that a trade-off has to be made between the percentage of user queries for which we can generate suggestions and the usefulness of those suggestions. Although our use case is not a social tagging website and the tags on `startpagina` do not form a folksonomy, we believe that our results do apply to search engines in these types of environments. Our results certainly invite us to investigate combinations of the models described in this paper with collaborative filtering methodologies that are currently being used to suggest queries or tags on social tagging websites.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A.: An optimization framework for query recommendation. In: WSDM 2010, pp. 161–170 (2010)
2. Baeza-Yates, R., Hurtado, C.A., Mendoza, M.: Query recommendation using query logs in search engines. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 588–596. Springer, Heidelberg (2004)
3. Bai, L., Guo, J., Cheng, X.: Query recommendation by modelling the query-flow graph. In: Salem, M.V.M., Shaalan, K., Oroumchian, F., Shakery, A., Khelalfa, H. (eds.) AIRS 2011. LNCS, vol. 7097, pp. 137–146. Springer, Heidelberg (2011)
4. Bai, L., Guo, J., Cheng, X., Geng, X., Du, P.: Exploring the query-flow graph with a mixture model for query recommendation. In: Proceedings of SIGIR Workshop on Query Representation and Understanding (2011)
5. Baraglia, R., Castillo, C., Donato, D., Nardini, F.M., Perego, R., Silvestri, F.: Aging effects on query flow graphs for query suggestion. In: CIKM 2009, pp. 1947–1950 (2009)
6. Beeferman, D., Berger, A.L.: Agglomerative clustering of a search engine query log. In: SIGKDD 2000, pp. 407–416 (2000)
7. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The query-flow graph: model and applications. In: CIKM 2008, pp. 609–618 (2008)
8. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: Query suggestions using query-flow graphs. In: Proceedings of the 2009 Workshop on Web Search Click Data, pp. 56–63 (2009)

9. Boldi, P., Bonchi, F., Castillo, C., Vigna, S.: From "dango" to "japanese cakes": query reformulation models and patterns. In: WI 2009, pp.183–190 (2009)
10. Bonchi, F., Perego, R., Silvestri, F., Vahabi, H., Venturini, R.: Efficient query recommendations in the long tail via center-piece subgraphs. In: SIGIR 2012, pp. 345–354 (2012)
11. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer networks and ISDN systems **30**(1), 107–117 (1998)
12. Bruza, P., Dennis, S.: Query reformulation on the internet: empirical data and the hyperindex search engine. In: RIAO 1997 (1997)
13. Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., Li, H.: Context-aware query suggestion by mining click-through and session data. In: SIGKDD 2008, pp. 875–883 (2008)
14. Craswell, N., Szummer, M.: Random walks on the click graph. In: SIGIR 2007, pp. 239–246 (2007)
15. Gayo-Avello, D.: A survey on session detection methods in query logs and a proposal for future evaluation. Inf. Sci. **179**(12), 1822–1843 (2009)
16. Hagen, M., Gomoll, J., Beyer, A., Stein, B.: From search session detection to search mission detection. In: OAIR
17. Hagen, M., Stein, B., Rüb, T.: Query session detection as a cascade. In: CIKM 2011, pp. 147–152 (2011)
18. Huang, C.-K., Chien, L.-F., Oyang, Y.-J.: Relevant term suggestion in interactive web search based on contextual information in query session logs. JASIST **54**(7), 638–649 (2003)
19. Huang, J., Efthimiadis, E.N.: Analyzing and evaluating query reformulation strategies in web search logs. In: CIKM 2009 (2009)
20. Lau, T., Horvitz, E.: Patterns of search: analyzing and modeling web query refinement. In: UMAP 1999, pp. 119–128 (1999)
21. Mei, Q., Zhou, D., Church, K.W.: Query suggestion using hitting time. In: CIKM 2008, pp. 469–478 (2008)
22. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web (1999)
23. Song, Y., Zhou, D., He, L.-w.: Query suggestion by constructing term-transition graphs. In: WSDM 2012, pp. 353–362 (2012)
24. Szpektor, I., Gionis, A., Maarek, Y.: Improving recommendation for long-tail queries via templates. In: WWW 2011, pp. 47–56 (2011)
25. White, R.W., Dumais, S.T.: Characterizing and predicting search engine switching behavior. In: CIKM 2009, pp. 87–96 (2009)